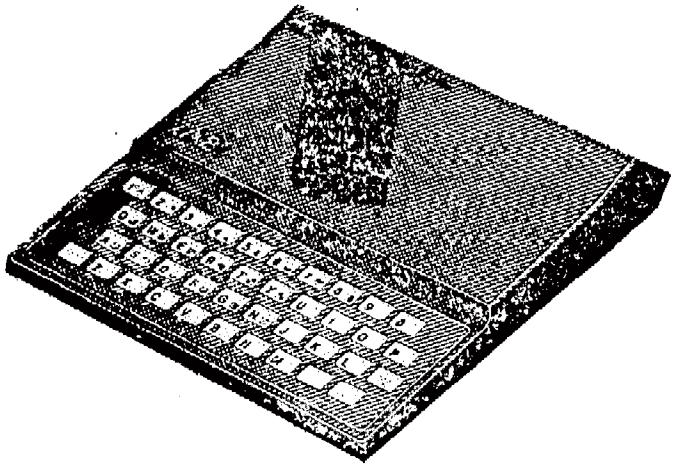
SINC-LINK



TIMEX-SINCLAIR USERS CLUB NEWSLETTER

Toronto, Ontario

EXECUTIVE OFFICERS

PRESIDENT:	George Chambers (416) 751-7559
LIBRARIAN:	Martin Mauk - tapes John Burns - magazines
ASS'T LIBRARIANS:	D. Cooper
	T. Haller
TREASURER:	Greg Lloyd
NEWS EDITORS:	John Roach (416) 447-7784
ACTIVITY DIRECTOR:	Cameron Hayne
LIAISON OFFICER (Out-of-town members):	George Chambers
	Mike Mitchell

LETTER FROM THE PRESIDENT

Over the past couple of months we have had quite a few new members join our club. I am sure they hope to benefit from association with other T/S users, and often their hopes are realized. However, in the rush of things, I am sure some of them tend to get lost in the shuffle, and ultimately lost to the club.

I would like to take this opportunity to welcome each and every one of you to our club. Use it, ask questions at meetings, borrow from our library, read the newsletter. Not the least important, contribute to the club activities. From my experience, you will profit immensely from the give and take of a more active role in the club. The invitation is extended; seize it. Cameron Hayne has done this, and has agreed to be our new Activities Director. He will be responsible for providing the demo's, talks, etc. at each meeting. Welcome to the executive, Cameron. We're pleased to have you.

Yours in computing
George Chambers
(President)

NOTES FROM THE EDITOR

If you have an article you would like published in the newsletter, give it to any member of the executive during the meeting or mail it to SINC-LINK c/o J. Roach, 56 Pachino Blvd., Scarborough, Ontario M1R 4J7.

Dan Sauve reports that nobody showed up at the 2 get-togethers he called to form a hardware SIG. He's still enthused - call him at 251-5623 if you're interested.

SYNCBITS

Ian Robertson

UPDATE FROM LAST ISSUE: It appears that someone is working on a TS2068/SPECTRUM interface. Apparently it is a lot more complicated than simply changing bus configuration. Aerco (Austin Texas) is marketing a great looking TS2068 Disc System. Bottom line is $508.00 (US) for this complete 2 drive system with cabinet and 64k Ram/8k Rom onboard. For details see their advert. in latest issue (#11) of T-S Horizons.

U.K. NEWS: Two interesting items gleaned from recent issues of *Your Computer*; a) a survey of 23,000 Open University students with home micros indicated that Spectrum was no. 2 with 15% and that the ZX81 was no. 3 with 11% (of further interest was the face that the CBM 64 was 4th with 6% and the Apple was tied with the TI99 at 4%), b) a comparison between the BBC and the QL which makes us dedicated Sinclair types cringe a little, as the BBC comes out ahead.

TS2068: There are several enterprising programmers advertising enhancements for the TS2050 Modem MTERM software which received such bad reviews because of the Documentation (or rather,the lack of it). Did you know that the Spectrum version of Tasword II does NOT WORK with the TS2068 with Spectrum Rom. It does not print out! Conversely, during a conversation with 21st Century Electronics I was advised that a TS2068 program which worked with the Spectrum Rom (inside a TS2068) did not work on the Spectrum. Verrry interesting.

TS 1000: If there still are some dedicated users out there, I suggest that if you want to purchase hardware items still advertised, that you get cracking. They will not be around forever. There are quite a few items available in the U.K. or from Integrated Data Systems (who carries most of the U.K. items). John Oliger is still offering his Hi-Res with colour kit.

SPECTRUM ROM: Of all the games reviewed I must say that SABRE WULF by Ultimate is the best (in my opinion that is). Two others deserve honourable mention, Jet Set Willy and Kokotoni Wilf. All three make good use of graphic capabilities. The following languages are available for the Spectrum; Forth (several versions), Beta Basic (enhanced Basic), Pascal, C, Micro Prolog, Logo and Lisp.

MAIL ORDER FROM U.K.: Most U.K. prices quoted in magazines are "Inclusive" or "Inclusive of VAT". VAT is a value added tax of 13% on top of the item price. Therefore deduct VAT from the quoted "Inclusive" price (multiplier .885) and add appropriate postage. On cassettes add approx. £2.00 each, to a total of 4, then add £0.75 for each additional. For larger items always add a min. of £5.00 to £7.50. Airmail costs are incredible, i.e., £7.50 for a small item (if larger than standard envelope size).

COMPUTERS AREN'T HUMAN

Most humans use the *decimal system*, which consists of ten digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), because humans have ten fingers. The computer does not have fingers, so it prefers other number systems instead. Here they are....

BINARY

Look at these powers of 2:

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$

Now try an experiment: pick your favorite positive integer, and try to write it as a sum of powers of 2. For example, suppose you pick 45. You can write it as 32+8+4+1. Suppose you pick 74; you can write it as 64+8+2. Suppose you pick 77; you can write it as 64+8+4+1. *Every* positive integer can be written as a sum of powers of 2.

Let's put those examples in a table:

ORIGINAL NUMBER	WRITTEN AS A SUM OF POWERS OF 2	DOES THE SUM CONTAIN... 64?	32?	16?	8?	4?	2?	1?
45	32+8+4+1	no	yes	no	yes	yes	no	yes
74	64+8+2	yes	no	no	yes	no	yes	no
77	64+8+4+1	yes	no	no	yes	yes	no	yes

To write those numbers in the *binary system*, replace "no" by 0 and "yes" by 1:

DECIMAL SYSTEM	BINARY SYSTEM
45	0101101 (or simply 101101)
74	1001010
77	1001101

The *decimal system* uses the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 and uses these columns:

thousands hundreds tens units

For example, the decimal number 7105 means "7 thousands + 1 hundred + 0 tens + 5 units".

The *binary system* uses only the digits 0 and 1, and uses these columns:

sixty-fours thirty-twos sixteens eights fours twos units

For example, the binary number 1001101 means "1 sixty-four + 0 thirty-twos + 0 sixteens + 1 eight + 1 four + 0 twos + 1 unit"; in other words, it means seventy-seven.

In elementary school, you were taught how to do arithmetic in the decimal system. You had to memorize the addition and multiplication tables:

DECIMAL ADDITION

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

DECIMAL MULTIPLICATION

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

In the binary system, the only digits are 0 and 1, so the tables are briefer:

BINARY ADDITION

	0	1
0	0	1
1	1	10 ← because two is written "10" in binary

BINARY MULTIPLICATION

	0	1
0	0	0
1	0	1

If society had adopted the binary system instead of the decimal system, you would have been spared many hours of memorization!

Usually, when you ask the computer to perform a computation, it converts your numbers from the decimal system to the binary system, performs the computation by using the binary addition and multiplication tables, and then converts the answer from the binary system to the decimal system, so you can read it. For example, if you ask the computer to print 45+74, it will do this:

```
 45 converted to binary is   101101
+74 converted to binary is +1001010
                           1110111   converted to decimal is 119
                           ↑
                           because 1+1=10
```

The conversion from decimal to binary and then back to decimal is slow. But the computation itself (in this case, addition) is quick, since the binary addition table is so simple. The only times the computer must convert is during input (decimal to binary) and output (binary to decimal). The rest of the execution is performed quickly, entirely in binary.

You know fractions can be written in the decimal system, by using these columns:

units point tenths hundredths thousandths

For example, $1\frac{5}{8}$ can be written as 1.625, which means "1 unit + 6 tenths + 2 hundredths + 5 thousandths".

To write fractions in the binary system, use these columns instead:

units point halves fourths eighths

For example, $1\frac{5}{8}$ is written in binary as 1.101, which means "1 unit + 1 half + 0 fourths + 1 eighth".

You know $\frac{1}{3}$ is written in the decimal system as 0.3333333..., which unfortunately never terminates. In the binary system, the situation is no better: $\frac{1}{3}$ is written as 0.010101.... Since the computer stores only a finite number of digits, it cannot store $\frac{1}{3}$ accurately—it stores only an approximation.

A more distressing example is $\frac{1}{5}$. In the decimal system, it's .2, but in the binary system it's .0011001100110011.... So the computer can't handle $\frac{1}{5}$ accurately, even though a human can.

Suppose you enter this on your ZX-81 computer:

```
PRINT "MY FAVORITE NUMBER IS ";4.001-4
```

The computer will try to convert 4.001 to binary. Unfortunately, it can't be converted exactly; the computer's binary approximation of it is slightly too small. The computer's final answer to 4.001-4 is therefore slightly less than the correct answer. Instead of printing MY FAVORITE NUMBER IS .001, the computer will print MY FAVOURATE NUMBER IS .0010000002 . Other computers come up with numbers like .000999987 .

To test your ZX-81's accuracy, try 4.0001-4, and 4.00001-4, and 4.000001-4 etc. you might be surprised at its answers.

M A K I N G B A C K - U P S

Fred Schakel
Timex/Sinclair Club
London, Ontario

All 2068 and Spectrum programs are saved/recorded on tape in one or more sections. These sections can be:

BASIC (program)
SCREEN$ (bytes)
or
MACHINE CODE (bytes)

Basic programs usually just have a basic section but machine code programs have 2 or 3 sections. These could include the basic section, maybe a screen$, and at least one code section.

HEADER

Every section on tape is preceeded by a header. This is the short part that loads when the border shows blue & red stripes. After a header has loaded, the title of the section will usually appear on screen. Besides the title, the header also loads in the length and starting address of the section.

FIGURE 1

```
  10 REM this is a sample
program to illustrate its
structure.

  20 LOAD "s" SCREEN$

  30 LOAD "mc" CODE

  40 CLS :PRINT "Stop Tape"

  50 etc.,etc.......

  60    (Basic Program)

  70

  80 STOP

  90 SAVE "example" LINE 20:
SAVE "s" SCREEN$: SAVE "mc"CODE

 100 STOP
```

PROGRAM STRUCTURE

Figure 1 shows some of the basic section of a program. All of the program would be on tape as illustrated in Figure 2. This is a typical 3-section program with basic, screen and code. Note line 90 which is the save line. All programs do not always have a save line. In this one, the basic is saved by

SAVE "example" LINE 20

This means that when the saved program is loaded, it will automatically go to line 20. Therefore line 20 is called the auto-start line of the program. Line 20 instructs to LOAD "s" SCREEN$ so that the screen will load while the tape is still running. Then line 30 will LOAD "mc" which are the machine code bytes. This is the last section to be loaded so line 40 PRINTS a message to stop the tape player.

The rest of the program is then executed and somewhere in the basic there will be an instruction to access the machine code that was previously loaded.(This is done with the USR command). As the header of each section was loaded, the screen would have displayed Figure 3, however, a SCREEN$ would cover up part of it.

FIGURE 2

```
Program header
Basic program
Screen header
Display screen
Machine code header
Bytes of code
```

FIGURE 3

```
program: example

bytes: s

bytes: mc
```

PROBLEMS

Sometimes a header won't display the title or the basic will not contain a SAVE line and you will not know where the auto-start line is.

To solve all these problems, we use a special utility program that allows us to load a header on top of it and it gives us all the required information. One such utility is called "Header". To use it we first load it in and just follow the directions on the screen. The header of each section is loaded in with "Header" and we can read the needed info on the screen and/or let the printer put it on paper.

LET'S TRY IT

1. We will make a back-up of the program in Figure 1. The first step is to obtain the following information:

   *Name of each section,
   *Auto-start line of basic,
   *Start address of each code section, and
   *Length of each code section

   If this cannot be readily obtained by looking for the save line in the basic, then you must use a special utility such as "Header".

2. Now load in only the basic section of the program in one of two ways. One way is to LOAD "" and BREAK when it has loaded the first section In some programs, you may not be able to BREAK. For these you must load them using: MERGE "". This will cause a break automatically at the end of the first section. Try to stop the tape player right at the end of this section so that it is positioned at the start of the second section.

3. This basic can now be saved onto your back-up tape by:

   SAVE "example" LINE 20

4. Now load the computer with the SCREEN$ section using one of the following:

   LOAD "" SCREEN$

   LOAD "" CODE

   Do not erase this from your computer screen. It will end the load with message: Immediately stop the tape at this point.

5. Now save the displayed screen onto the back-up tape using:

   SAVE "s" SCREEN$

   NOTE a SCREEN$ is always put at address 16384 and it is always 6912 bytes long. Therefore we could have saved it using:

   SAVE "s" CODE 16384,6912

6. LOAD in the last section (machine code) from the original tape with:

   LOAD "" CODE

7. Save this code and indicate the address it is to be put and the length of it.

   SAVE "mc" CODE 40000,1200

   In the sample the 40000 is the starting address of the code and 1200 is the number of bytes long it is. These must be included whenever saving code.

7. If the program has more sections just repeat the procedure for the extras, and if there are fewer, omit those. Now you may wish to verify everything using:

   VERIFY ""

   to check the basic and:

   VERIFY "" CODE

   to check the code. You can not verify the SCREEN$ so just skip over it when verifying.

PROGRAMMING TRICKS

The title of the basic program will always print on screen when its header is loaded, but the title of any following sections is sometimes not shown. This is done by making the ink and paper the same colour before that section is loaded. Obtain the hidden title by the use of "Header".

When the basic section is loaded and you wish to BREAK the loading process at the end of the section, you may get a crash or it may keep on loading. This trick is done by a POKE in the auto-start line or by the ON ERR command. Overcome this problem by loading the basic with the MERGE command.

Sometimes there is a blank area of tape between sections. This would be needed for the basic to do something before the line that loads the next section. For example, the "Upload 2000" program does a short display routine between the basic and the machine code. Be sure to insert this same length of blank in your back-up. You can determine this length by timing it or with the tape counter.

NOTES

1. Make back-ups using the same names as the original unless you are sure the program won't need names.
2. If you are in doubt whether the next section is a screen or machine code, load it with: LOAD "" CODE
3. Sometimes you will not need to use the Header utility. Many programs have all the necessary info in a save line in the basic section.
4. When saving a screen, it must be present on the TV screen or it will not save.
5. If the basic has a save line you can GOTO this line to save all sections but Note #4 applies. When saving more than one section by using the save line, be ready to press a key when prompted.
6. Note that throughout this article, I have used the term "back-up" as opposed to a "copy". Even though we are dealing with making a copy or duplicate of an original program, the term "back-up" has a more legal ring to it!

A HEADER PROGRAM FOR THE ZX81

When I got my TS2068 I thought one of the marvellous things about it was a Header program by which one could get the name of a program displayed on-screen a few seconds after it started loading. 'If only the ZX81 had that feature it would be marvellous',I thought.

Where have I been all this time? I've only just discovered there is a program to do practically the same thing. It is to be found in Mike Lord's book 'Explorers guide to the ZX81'. How I never spotted it I'll never know.

With the thought there must be many in the same position as myself, I would like to present this practical program.

It is a M/C routine that is placed in a REM statement at the beginning of a short Basic program. First create a REM statement of 66 spaces(or more) Then enter the lines as shown in Fig. 1. Put your computer in the FAST mode, RUN the program and enter the numbers in Fig. 2 starting with the top row ,left to right. When you are done check that the sum of the numbers is 7620. Then delete all lines except the line 1 containing the REM statement.

Enter the lines in Fig. 3. SAVE this program, since this is the program you will use to obtain HEADER information from your other program tapes.

When you RUN this program, it will prompt you to start playing the tape which you wish to identify. At the start of the LOADing of a tape the program name under which it was SAVEd will appear on the screen. Any other programs appearing on the tape will be similarly identified.

Fig. 1

```
  1 REM 123456789012345678901
2345678901234567890123456789012345
67890123456
 10 FOR I=16514 TO 16579
 20 INPUT A
 30 POKE I,A
 40 SCROLL
 50 PRINT I;" ";A
 60 NEXT I
 70 PRINT "FINISHED"
110 PRINT "PRESS A KEY TO CHECK
120 IF INKEY$="" THEN GOTO 120
130 LET TOTAL=0
140 FOR I=16514 TO 16579
150 LET X=PEEK I
160 LET TOTAL=TOTAL+X
170 NEXT I
180 PRINT TOTAL
```

Fig. 2

205	35	15	205	138	64
24	251	14	1	6	0
62	127	219	254	31	23
23	56	15	16	245	241
205	138	64	121	215	203
121	40	247	205	43	15
201	213	30	148	6	26
29	219	254	23	203	123
123	56	245	15	245	209
32	4	254	85	48	206
63	203	17	48	201	201

FIG 3

```
  1 REM LN 77LN ▒RND/ CLS ▒▒▒ ▒
▒<= RETURN 3**S( ( PRINT LET LN ▒
RND?NOT ACS ?C RUN LN F?TAN STR$
 2▒▒,1<= RETURN *ACS ??S PRINT (
 PRINT SGN 4▒ RETURN ?KEXP ZACS
)KTAN TAN
 20 RAND USR 16514
 30 PRINT AT 20,0;"PRESS A KEY
TO READ TAPE"
 40 IF INKEY$="" THEN GOTO 40
 50 CLS
 60 GOTO 20
100 SAVE "HEADER READE▒"
110 GOTO 30
```

G.F. CHAMBERS

** ERROR ** ERROR ** ERROR **
SEARS RGB MONITOR

In the article on the Sears RGB monitor in the last issue of the newsletter there was a reference to replacing the 330 ohm resistor with a 10K variable resistor. While the article was correct, the accompanying sketch had an arrow pointing (incorrectly) to the 3300 ohm resistor. It should be the 330 ohm resistor that is replaced.
Sorry about that!

USE OF THE 'IN' FUNCTION
By G F Chambers

It is sometimes desirable to be able to use two keys at the same time. Many games could be improved with this capability. On the TS2068 this is possible through the use of the 'IN' function. The following Table A, and Etch-a-Sketch demonstration program will illustrate this possibility.

TABLE A

<-----IN values----->

IN Address	30	29	27	23	15
32766	BR	SH	M	N	B
49150	ENTER	L	K	J	H
57342	P	O	I	U	Y
61438	0	9	8	7	6
63486	1	2	3	4	5
64510	Q	W	E	R	T
65022	A	S	D	F	G
65278	-	Z	X	C	V

```
  5 REM An ETCH-a-SKETCH type
       of drawing routine to
       illustrate the use of
       the IN function on
       the TS2068 computer.
  6 REM
  7 REM      By G F Chambers
  8 REM
  9 REM    Use keys 0,A,P,and Q
       for cursor control.
        Use keys 1-5 for
        choice of color.

 10 LET x=125: LET y=95
 20 IF IN 64510=30 AND y<172 TH
EN LET y=y+1
 30 IF IN 65022=30 AND y>1 THEN
 LET y=y-1
 40 IF IN 57342=30 AND x<254 TH
EN LET x=x+1
 50 IF IN 57342=29 AND x>1 THEN
 LET x=x-1
 55 IF IN 63486<>31 THEN GO SUB
 130: GO TO 20
 60 PLOT x,y
 120 GO TO 20
 130 LET r=0: LET c=32-IN 63486
 140 LET c=c/2: LET r=r+1: IF c<
=1 THEN GO TO 160
 150 GO TO 140
 160 IF INKEY$<>"" THEN GO TO 16
0
 170 INK r: RETURN
```

E R A S I N G T A P E S

What does it take to erase a tape? I like to use a bulk eraser but, how far away should it be kept from other tapes? After performing a number of experiments, I concluded that there is only one major concern for accidental erasure of programs. Following, is a list of the experiments which did not have any affect on a short pre-recorded program.

* Moving a head-demagnifier around the tape.
* Holding tape against a vacuum cleaner while it was turned off & on & running.
* Moving tape around electric drill & turning it off & on.
* Holding a small, 4" speaker-magnet against tape & moving it.
* Held 2" away from soldering iron for a minute.
* Held in contact with colour TV screen while it was turned off & on.
* Left tape under a telephone while it rang 3 times.
* Made 3 SCREEN COPIES on the 2040 printer with tape in contact with printer.
* Held tape about 1 (ONE) inch from bulk tape eraser while turning eraser off & on and moving tape around!!!

A larger speaker magnet (e.g.: 8 oz.) would probably erase the program. Likewise, moving the tape within 1" of the bulk eraser is "Goodbye Program", even if the eraser is switched on & off once. However, this is not considered accidental.

So why be concerned about erasing your favourite program? The real NUMBER ONE cause for lost programs is the careless finger on the record button.

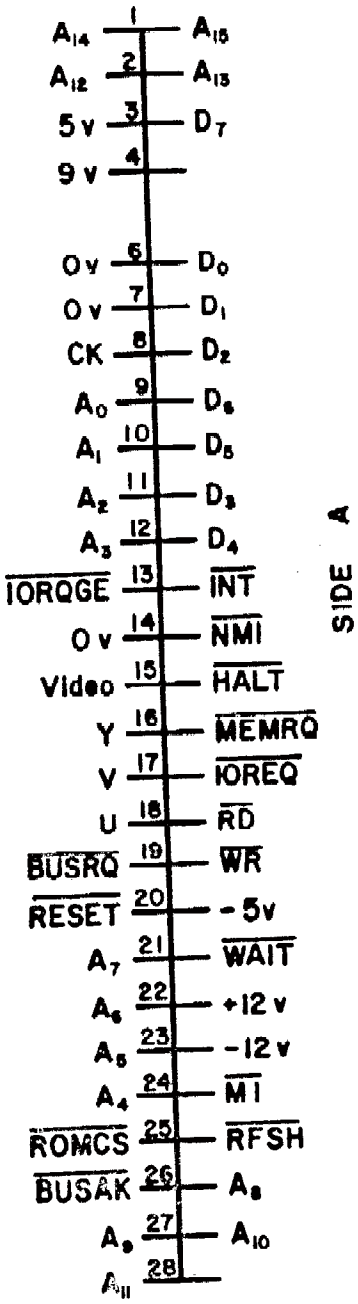
Here are a few precautions to take to prevent this from happening to you:

* Remove WRITE-ENABLE tabs on tapes. They can always be replaced with Scotch tape.
* Label the contents on every tape.
* Have back-ups, just in case.
* Think before recording.

Fred Schakel
London T/S Gr

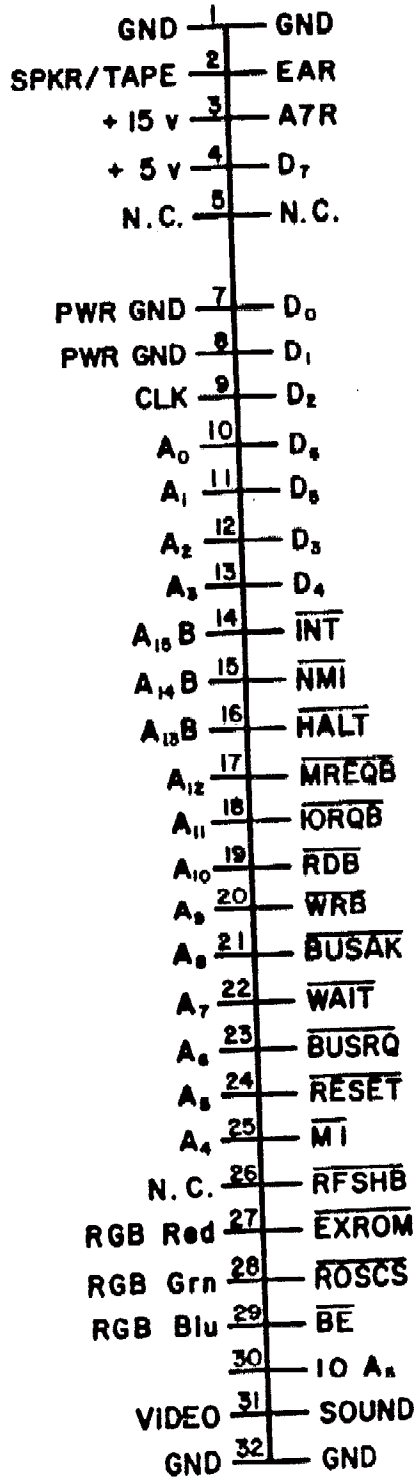
SPECTRUM

BOTTOM TOP



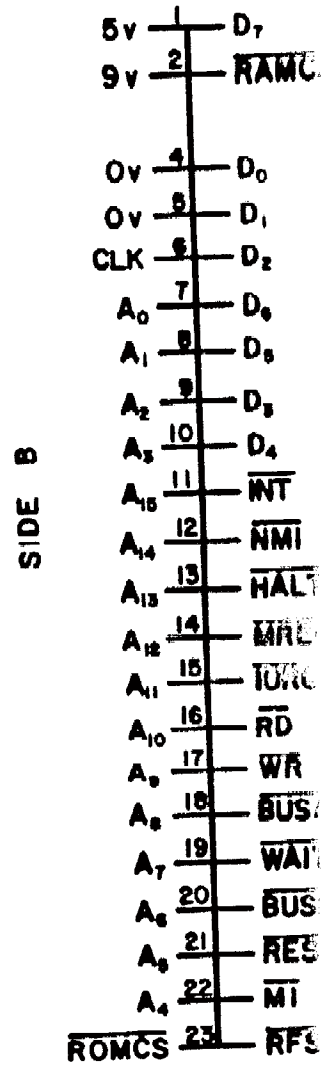
TS 2068

BOTTOM TOP



ZX-81

BOTTOM TOP



ZX81 Hardware News

by Peter McMullin for T.T.S.U.C. 2/85

#1 in a Series for the ADVENTUROUS, COMMITTED ZX81 Owner.

Many who took advantage of the great price on **GLADSTONE 64K RAM** packs have expressed the need to disable the 8K-16K block, so they can use some of the many peripherals which are mapped to that area. To this end, I have drawn an improved schematic of the **G64K** for reference, and written an article, with documented schematics, outlining the necessary procedures.

A simple address decoder and MREQ NOT gating does the trick, with options depending on required switching flexibility. In fact, although not mentioned in the article, this method could be used to selectively disable ANY block of memory between 0 & 64K!

The details are too involved to relate here, so I have placed copies of the project manual in the **CLUB LIBRARY:** it is available there for those who require it.

Apologies to those waiting for info on using the cheap but nifty **EXCELTRONIX 5" Greenscreen** monitor. (I just found the file!) The problem is with the supplied *sync separator*, **NOT** with the monitor itself. My solution involves using the partial sync-separation provided by the **Random Access** VIDEO INVERTER circuit, which was published as a construction project in Computers & Electronics, May '83 issue. As with the **G64K** project, I will place full details in the LIBRARY as soon as I can prepare the material.

I am composing this text on my **64K ZX81**, outfitted with a surplus **Apple(™)** clone keyboard, using **Word Sinc II.** Text is output on an **EPSON MX-80** printer, via the **EPROM SERVICES** Centronics interface and software driver, which I recently purchased from Integrated Data Systems.

It is encouraging to report that MEMOCALC, and ZX PRO-FILE(Centronics I/F version) both print out just fine with the EPROM SERVICES I/F and software. Output of the printer CTRL and <ESC> codes, so valuable in wordprocessing, is a sticky subject, though. I BREAK from WSII menu, & LPRINT A$ to do it here.

I am currently awaiting replies from both **P. Hargrave** and **EPROM SERVICES** regarding **WSII.4** compatibility. Things seem promising, so I'll keep you posted.

My final topic for this issue is the most exciting. I recently received documentation on the long-awaited **ZX81 Video Upgrade** project, created by **John Oliger, 11601 Whidbey Dr., Cumberland, IN 46229.** The following features should make you sit up and take notice:

1) SLOW mode is 5.5X normal speed, i.e. same as 2068. Some M/C games run TOO FAST, and arcade-speed games in BASIC become possible! FAST mode, as usual at 6X normal SLOW speed, is BLINK-FREE!
2) You get COLOUR VIDEO with UPPER and lower case on screen, as an alternative to inverse characters. Colour control is equiv. to 2068.
3) Machine code can now run anywhere in 64K, with free use of the IX register, and schemes using vectored hardware Interrupts may be supported.
4) System is completely transparent to existing software.
5) Display File CRASHES are NO MORE!

The project consists of two circuit boards, one for the **TMS9918A Video Display Processor**(which supports SPRITE graphics) and 16K Video RAM. The other circuit board holds an enhanced(i.e. display routines rewritten) version of the ZX81 ROM, which it overrides. A Motherboard or cable is required.

Mr. Oliger sells both boards, and all parts except the VDP and it s crystal, for $48.95 U.S. (Quick! Where's my checkbook?) Write to him for details.

Discoveries like these make the **AERCO Disk System** for the ZX81 seem more attractive all the time to a hardware addict like me. But that's for another discussion. *Happy Computing!*